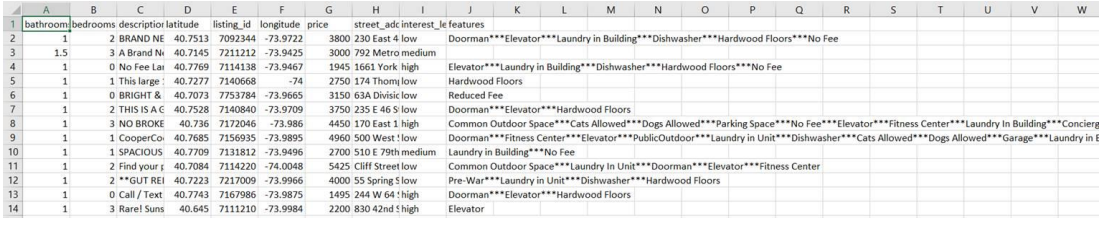
1. Data overview
    1.1 Data description
    1.2 Statistics & visualization (stat-457-project2-data-exploration.ipynb, feature_selection_0417.Rmd)
2. Data preprocessing
    2.1 Data cleaning (stat-457-project2-data-exploration.ipynb)
    2.2 Text analysis (Text tokenizer.ipynb)
3. Feature selection
    3.1 Chi-square test (stat-457-project2-data-exploration.ipynb)
    3.2 Variable importance (feature_selection_0417.Rmd)
    3.3 Boruta package (feature_selection_0417.Rmd)
4. Simple prediction (by R)
    4.1 XGB boost (XGboost_text_nopca.Rmd)
    4.2 Random forest (Random forest.Rmd)
    4.3 Boosting (boosting_noPCA.Rmd)
    4.4 Neural network (model_neural network.Rmd)
    4.5 Model averaging (XGboost_text_nopca.Rmd)
5. Summary
6. Reference

1. Data overview

1.1 Data description

From the data description on the website, we have information for apartment rental, and 8-186th columns are all values with 0 and 1. Since this group of columns with 0-1 includes most of the attributes, I will discuss separately for 1-7 columns and 8-186 columns.

From the picture for how the professor create 8-186 columns, I found on the last column of original dataset, there is different words, and extract different unique words as 8-186 columns, similar as one-hog encoder.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	bathroom	bedrooms	description	latitude	listing_id	longitude	price	street_adc	interest_le	features													
2	1	2	BRAND NE	40.7513	7092344	-73.9722	3800	230 East 4	low	Doorman***Elevator***Laundry in Building***Dishwasher***Hardwood Floors***No Fee													
3	1.5	3	A Brand Ne	40.7145	7211212	-73.9425	3000	792 Metro	medium														
4	1	0	No Fee La	40.7769	7114138	-73.9467	1945	1661 York	high	Elevator***Laundry in Building***Dishwasher***Hardwood Floors***No Fee													
5	1	1	This large	40.7277	7140668	-74	2750	174 Thom	low	Hardwood Floors													
6	1	0	BRIGHT &	40.7073	7753784	-73.9665	3150	63A Divisi	low	Reduced Fee													
7	1	2	THIS IS A G	40.7528	7140840	-73.9709	3750	235 E 46 S	low	Doorman***Elevator***Hardwood Floors													
8	1	3	NO BROKE	40.736	7172046	-73.986	4450	170 East 1	high	Common Outdoor Space***Cats Allowed***Dogs Allowed***Parking Space***No Fee***Elevator***Fitness Center***Laundry In Building***Concierg													
9	1	1	CooperCo	40.7685	7156935	-73.9895	4960	500 West !	low	Doorman***Fitness Center***Elevator***PublicOutdoor***Laundry in Unit***Dishwasher***Cats Allowed***Dogs Allowed***Garage***Laundry in E													
10	1	1	SPACIOUS	40.7709	7131812	-73.9496	2700	510 E 79th	medium	Laundry in Building***No Fee													
11	1	2	Find your p	40.7084	7114220	-74.0048	5425	Cliff Street	low	Common Outdoor Space***Laundry In Unit***Doorman***Elevator***Fitness Center													
12	1	2	**GUT REI	40.7223	7217009	-73.9966	4000	55 Spring S	low	Pre-War***Laundry in Unit***Dishwasher***Hardwood Floors													
13	1	0	Call / Text	40.7743	7167986	-73.9875	1495	244 W 64 !	high	Doorman***Elevator***Hardwood Floors													
14	1	3	Rare! Suns	40.645	7111210	-73.9984	2200	830 42nd !	high	Elevator													

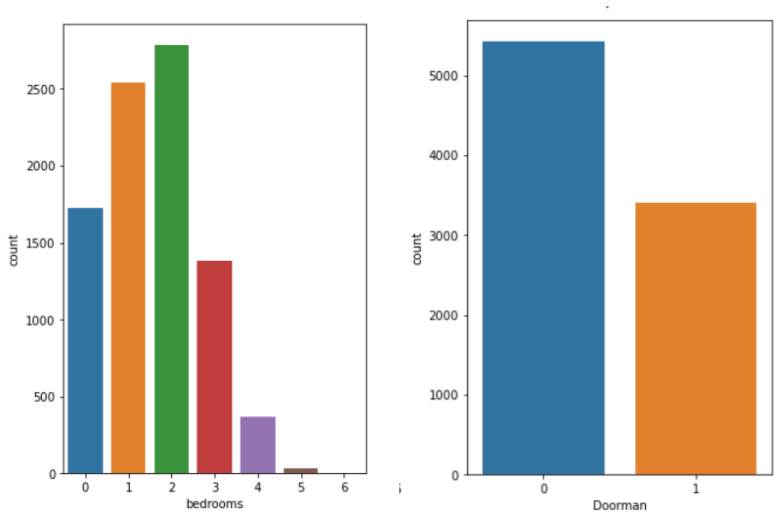
But the problem is, some words have similar meaning, or the same meaning. For example, 'high.ceilings' has the same meaning as 'high.ceiling', but the data says they are in different columns. I will clean the data in preprocessing steps.

1.2 Statistics & visualization

	interest_level	bathrooms	bedrooms	latitude	longitude	price	street_address	Doorman	Elevator	Laundry.in.Building	..
0	medium	2.0	4	40.7182	-74.0148	5354	41 River Terrace	1	1	1	..
1	low	1.0	2	40.7231	-73.9938	4695	232 Elizabeth Street	0	0	0	..
2	medium	1.0	1	40.7463	-73.9132	1850	39-89 51st St	0	0	0	..
3	medium	1.0	2	40.7349	-73.9818	3030	324 E 19 Street	0	0	0	..
4	high	1.0	1	40.7031	-73.8119	1500	88-40 144th St	0	0	0	..

First, look at the head of data. I convert interest_level to 0,1,2 (low, medium, high). For street address, I will do text analysis later, now just remove it and store the information for this column for further use. Then look at continuous variables, latitude, and price. Since they are continuous variables and have different ranges, we may need a scaler to deal with that. I'm using min-max scaler for latitude, longitude and price.

Here is histogram for bedrooms and Doorman:



They are evenly distributed.

Found that latitude and longitude are close correlate with each other. Draw a scatter plot between these two covariates:
(scatter plot)

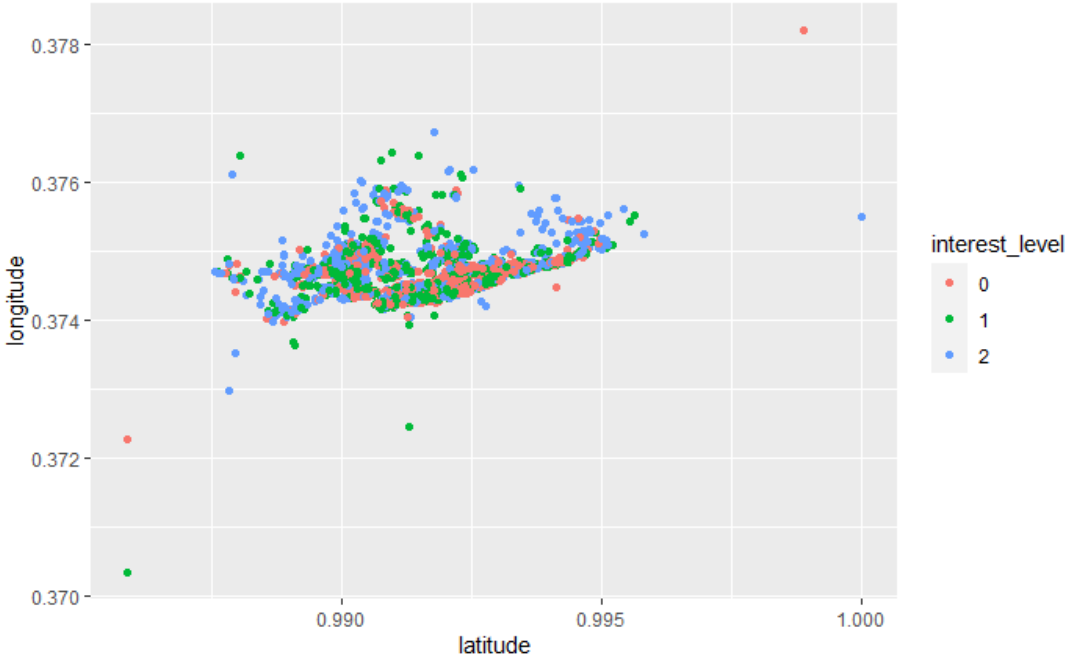
From the scatter plot, some continuous features may have outliers. Check skewness and draw boxplot:

skewness for continous columns

```{r}
library(e1071)
for(i in 4:6){
  print(skewness(trainDat[,i]))}
```

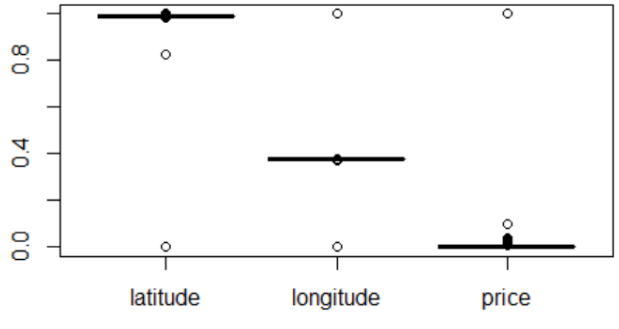
```
[1] -89.32235
[1] 46.47839
[1] 89.44321
```

There is two columns has large skewness , which is not good. The range(-0.5,0.5) is acceptable for the skewness.

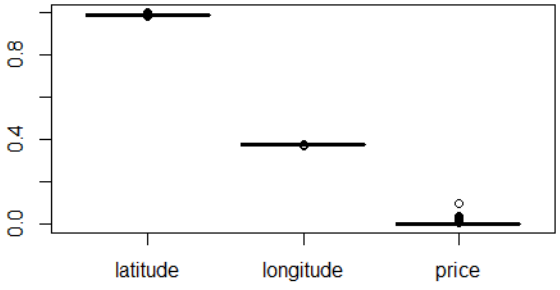


Here is the scatter plot, shows the interest level on different longitude and latitude. I choose to detect the points outside this graph's range as outliers. Different interest levels are evenly distributed in different longitude and latitude.

Then, apply minmax scaler to make sure each column has the same range. Draw boxplot to detect outliers.



All columns have outliers. Delete these data points.



This one looks better.

```
[1] -0.022284
[1] 1.283506
[1] 16.01148
```

The skewness for each column is better than before.

Then, check 8th – 108th columns' distribution. Since they all have values 0 and 1, check if they are sparse data. Delete columns when one of 0 and 1 is above 95%.

```
# check sparsed data, one column > 95% means sparsed
sparsed_1 = sum_each_col[sum_each_col < len(train) * 0.05]
sparsed_1
```

```
Reduced.Fee             216
Common.Outdoor.Space    245
Parking.Space            97
Concierge                21
PublicOutdoor            46
                        ...
Gym.In.Building           3
Decorative.Fireplace      3
Central.AC                4
Sundeck                   3
Yoga.Classes              3
Length: 160, dtype: int64
```

```
sum_each_col[sum_each_col > len(train) * 0.95]
```

```
Series([], dtype: int64)
```

So, they are many columns has more than 95% value 0, and no columns has more than 95% value 1. We need to delete sparse columns after data cleaning.

2. Data preprocessing

From the above analysis, I found that our dataset has duplicated columns, or columns with the similar meanings. We found there is many sparse columns that need to be deleted after cleaning the data.

2.1 data cleaning

I'm cleaning the dataset, by remove some of the sparse data first, since there are too many columns for the cleaning process.

```
sum_each_col = train_subset.sum(axis=0)
sparsed_1 = sum_each_col[sum_each_col < 50]
sparsed_1.sort_values(ascending=False)
```

```
publicoutdoor             46
light                     43
high.ceiling              42
granite.kitchen           40
laundry                   38
                          ..
assigned.parking.space     2
all.utilities.included     2
wifi                       2
x24.7.concierge            2
on.site.parking            2
Length: 100, dtype: int64
```

Delete all columns that has occurrence < 50.

Then, see the remaining columns and find the columns with similar meanings, such as merging 'roof.deck' and roof deck', merging 'laundry.in.building' and 'washer.dryer.in.building'.

```
train_subset['common.outdoor.space'] = train_subset_copy['common.outdoor.space'] + train_subset_copy['common.backyard'] +
train_subset_copy['common.garden']
train_subset['fitness.center'] = train_subset['fitness.center'] + train_subset_copy['gym']+ train_subset_copy['gym.fitness'] +
train_subset_copy['gym.in.building'] + train_subset_copy['health.club']
train_subset['hardwood'] = train_subset['hardwood.floors'] + train_subset['hardwood']
train_subset[ 'high.ceilings'] = train_subset[ 'high.ceilings'] + train_subset_copy['high.ceiling']
train_subset['high.speed.internet'] = train_subset_copy['high.speed.internet'] + train_subset_copy['wifi']
+train_subset_copy['wifi.access']
train_subset['live.in.super'] = train_subset['live.in.super'] + train_subset_copy['live.in.superintendent']
train_subset['laundry.room'] = train_subset_copy['laundry.room'] + train_subset_copy['laundry'] +
train_subset_copy['washer...dryer'] + train_subset_copy['washer.dryer']
train_subset['laundry.in.building'] = train_subset['laundry.in.building'] + train_subset_copy['washer.dryer.in.building'] +
train_subset_copy['on.site.laundry']
train_subset['laundry.in.unit'] = train_subset['laundry.in.unit'] + train_subset_copy['washer.dryer.in.unit']
train_subset['pre.war'] = train_subset['pre.war']+ train_subset['prewar']
train_subset['roof.deck'] = train_subset['roof.deck'] + train_subset_copy['roofdeck']
train_subset['parking.space'] = train_subset['parking.space'] + train_subset['garage'] +
train_subset_copy['common.parking.garage']+ train_subset_copy['full.service.garage']
```

Also do the same step with testing data.

After that, most columns have occurrence > 100.

2.2 text analysis

Since we have another column, address, which is hard to get useful information. I'm using keras package, text.Tokenizer function, by turning each text into either a sequence of integers into word counts. By this method I generated 10000 columns for the word count for the address column. Then, I use dimensional reduction method, PCA, reduce the

dimension and select all columns with explained variance ratio > 0.1 and add to my original dataset.

3. Feature selection

3.1 chi-square test

For these selected columns, I want to check if two categorical variables are independent. This can be done with Chi-Squared test of independence. By python sklearn select K best package, I found there is two columns have p value > 0.05, delete them, since these two columns are not statistically significant. All other features satisfy the goodness of fit.

3.2 variable importance

By using caret package, I'm selecting variables by variable importance, using a tree method 'rpart'. RPART implement the Recursive Partitioning and regression Tree, it recursively creates a decision tree that strives to correctly classify menbers of the population by splitting it into sub-populations, based on several dichotomous independent variables. [1]

I wrote the code to select several top important features: (rpart prints top 20 features)

	Overall <dbl>
price	100.00000
no.fee	60.47749
hardwood	59.91909
dishwasher	35.67063
X1	21.30742
longitude	20.13509
reduced.fee	0.00000
fitness.center	0.00000

So, the first 6 features are selected by rpart.

3.3 Boruta package

Boruta package tries to capture all the important features with respect to an outcome variable. First, it duplicates the dataset, and shuffle the values in each column. Then, it trains Random Forest Classifier on the dataset. At last, the algorithm checks for each of the features if they have higher importance. At every iteration, the algorithm compares the Z-scores of the shuffled copies of the features and the original features to see if there are improvements.

By this method, I selected these features: factor(interest_level) ~ bathrooms + bedrooms + latitude + longitude + balcony + cats.allowed + common.outdoor.space + dining.room + dishwasher + dogs.allowed + doorman + elevator + exclusive + fitness.center + furnished + garage + garden.patio + hardwood + high.ceilings + high.speed.internet + laundry.in.building + laundry.in.unit + laundry.room + live.in.super + loft + lowrise + new.construction + no.fee + outdoor.space + parking.space + pre.war + private.outdoor.space + reduced.fee + renovated + roof.deck + simplex + price

By combine the result of both methods, I get my final selected features.

4. Prediction models.
I'm trying to use XGBoost, Boosting, Neural network and random forest for this part. I tried the raw dataset first, then use the final dataset on the best models and show the best score.

4.1 XGBoost
Since XGboost works well in project1, I'm try this first. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. XGBoost library implements the gradient boosting decision tree algorithm. Gradient boosting is a type of machine learning boosting. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error.[2]

I also use hyper pruning parameter for choose the best parameter:

eta: learning rate. Eta shrinks the feature weights to make the boosting process more conservative. My choice: c(0.01,0.05,0.1)
Gamma: Minimum loss reduction required to make a further partition on a leaf node of the tree. My pruning choice: c(0.1, 1,2,3)
Max_depth: max depth of the tree. Increase this value will let the model become more complex My choice: c(3,4,5,6)
Min_child_weight: Minimum sum of instance weight (hessian) needed in a child. The larger the value is, the more conservative the algorithm will be. My choice: c(3,4,5,6)

Final parameters are: booster = "gbtree", eta = "0.01", min_child_weight = "0.05", max_depth = "0.1", gamma = "3", subsample = "5", colsample_bytree = "0.1", objective = "1", eval_metric = "2", num_class = "3", validate_parameters =

"0.75", booster = "1", eta = "multi:softprob", min_child_weight = "mlogloss", max_depth = "3", gamma = "TRUE"

My final score for XGboost is 0.84175.

4.2 Random forest

Random forest is based on the bagging algorithm and uses an ensemble learning technique. It grows the decision tree to full size and combines the output of all the trees. So, it can reduce the overfitting problem and minimize the variance. I use hyper parameter method to choose parameter mtr, number of variables available for splitting at each tree node.

The final model is: mtry = 7

My score for random forest is 0.90902. (by raw dataset, it looks like there is other better choices)

4.3 Boosting (gradient boost)

Gradient boosting uses the average model and decision tree as estimators, and it is derived from the weights optimized by gradient descent to minimize the cost function.

I'm use caret package, use train control as method cv, fold number = 10, to hyper pruning the parameter with interaction depth (number of splits it has to perform on a tree), n.tree (number of tree), n.minobsinnode (split each node until there is only n observation in each terminal node)

The final values used for the model were n.trees = 200, interaction.depth = 4, shrinkage = 0.05 and n.minobsinnode = 12.

My final score for boosting is 0.83833.

4.4 Neural network

The neural network system is: first, analyze the input data and detects the properties of the inputs, layer by layer, and gives the output. Then the network system adjusts its internal weightings to the answers. The adjustment helps in improving its performance.

I'm using caret package, neuralnet function, use the hyper parameter to select size and decay. Size is the number of units in hidden layer and decay is the regularization parameter to avoid over-fitting.
The final choice for the model is: size = 5, decay = 0.5

My score for neural network is 0.88031.

4.5 Model averaging

Compare the above models, I found that Xgboost and Boosting methods are the best, which has score within 0.85. So, I want to combine the result from Xgboost and Boosting to get the optimal result. By average the model result from both methods, it can reduce the bias when two models predict completely different results.

By take the average of Xgboost and Boosting, the final score is:

So, model averaging improves my result a lot.

5. Summary

In summary, the model averaging with Xgboost and Boosting gives me the best result. From exploring the data, I found the dataset need to be cleaned, found the outliers, and check the distribution of each columns. From data preprocessing I cleaned the data and get useful information from text analysis. Then I choose the best features from Chi-square, rpart(decision tree) and Boruta package. At last, I select different models to decide which one did the best, and use model averaging to make the best prediction.

I also tried to use PCA to reduce the dimension (for column 8-186) because there is 100+ columns. But the result shows it does not make improvement to my prediction.

Compare with original dataset, when I use XGboost, the score increases from 0.852 to 0.841, which is a great improvement on the dataset. The initial model, logistic regression runs a score about 1, the total improvement in my result is 1->0.826, about 0.174.

For the further improvement, I can try advanced deep learning with CNN or RNN to predict the result. The size of data is suitable to use this method.

6. References:

[1] https://en.wikipedia.org/wiki/Recursive_partitioning#:~:text=Recursive%20partitioning%20is%20a%20statistical,on%20several%20dichotomous%20independent%20variables

[2] https://www.displayr.com/gradient-boosting-the-coolest-kid-on-the-machine-learning-block/